IFW

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | | |
|---|---|---|---|
| Applicant: | Charles M. Potter; Joseph Zelenka; Donald Moffat | | |
| Serial No.: | 10/783,999 | | |
| Filed: | February 20, 2004 | Customer No.: | 28863 |
| Examiner: | Unknown | | |
| Group Art Unit: | Unknown | | |
| Docket No.: | 1028-025US01 | | |
| Title: | DIMENSION-BASED PARTITIONED CUBE | | |

CERTIFICATE UNDER 37 CFR 1.8: I hereby certify that this correspondence is being deposited with the United States Post Service, as First Class Mail, in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on May 11, 2004.

By: Beth M. Lindblom
Name: Beth M. Lindblom

**SUBMISSION OF CERTIFIED COPIES OF ORIGINAL FOREIGN APPLICATIONS**

Commissioner for Patents
Alexandria, VA 22313-1450

Dear Sir:

This application claims priority under Title 35, United States Code, Section 119, to Canadian Application No. 2,419,502, filed on February 21, 2003.

A Certified copy of the above-identified earlier-filed Canadian application is enclosed.

Respectfully submitted,

Date: May 11, 2004

By: Kent J. Sieffert
Reg. No.: 41,312

Shumaker & Sieffert, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Phone: (651) 735-1100
Fax: (651) 735-1102

Office de la propriété intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An Agency of Industry Canada

# *Bureau canadien des brevets*

*Certification*

# *Canadian Patent Office*

*Certification*

La présente atteste que les documents ci-joints, dont la liste figure ci-dessous, sont des copies authentiques des documents déposés au Bureau des brevets.

This is to certify that the documents attached hereto and identified below are true copies of the documents on file in the Patent Office.

Specification and Drawings, as originally filed, with Application for Patent Serial No: **2,419,502**, on February 21, 2003, by **COGNOS INCORPORATED**, assignee of Charles Mike Potter, Joseph Zelenka and Donald Moffat, for "Time-Based Partitioned Cube".

Agent certificateur/Certifying Officer

February 20, 2004
Date

Canada

(CIPO 68)
04-09-02

OPIC CIPO

## Abstract

A system for storing data is provided. The system comprises one or more member cubes for storing time-partitioned data, and a control cube for accessing the member cubes.

# Time-Based Partitioned Cube

## FIELD OF THE INVENTION

The invention relates generally to software and databases, and in particular to a time-based partitioned cube.

## BACKGROUND OF THE INVENTION

Online analytical processing (OLAP) is a growing application area of information technology (IT). The data subject to OLAP analysis is typically stored either in a relational online analytical processing (ROLAP) database or, more often, in a data structure that has been designed specifically for this purpose - a multidimensional online analytical processing (MOLAP) database, i.e., a cube.

ROLAP handles theoretically unlimited volume of data, but the analysis is generally slow. Cubes have better performance, but, typically, the size of a cube is subject to limitations due to intrinsic constraints. The performance of a cube arises from its design: cubes are optimized for fast access rather than for ease of their creation and update. During initial cube creation and when it is being maintained, the data is analyzed and structures created to enable fast access of selected subsets of the data in any of its business dimensions (or combination of dimensions).

In most cases, the data to be analyzed is not all available at the time the cube is created. Typically, the data arises at regular intervals and must be added to the existing cube. As indicated above, cubes are ill suited for growing data volume in time. There are two aspects to this – rigidity of internal data structures and lack of support of changing data attributes in time (i.e., slowly changing dimensions). More specifically,

(a) The internal structures in cubes were optimized for fast access of the original data set. When adding data to the cube, updating these structures is slow and inefficient, and the resulting structures may not be optimal for accessing the total set of data in the new cube. Consequently, as the cube complexity grows in time and volume, its performance deteriorates.

(b) Data attributes (i.e., metadata) stored in the cube are static. Even though they evolve in time (e.g., products are introduced / discontinued, corporation structure and hierarchy is re-organized, staff moves in, out and within the organization, new sales channels appear, etc.) the same multi-dimensional

metadata structure is applied to all time intervals, regardless of whether or not all attributes apply in the given time interval. This increases the volume of information noise in analytic results and reports that makes their interpretation increasingly harder with progress of time.

The problem of performance of updating an existing cube and accessing the resulting cube has not been solved in the past. Support of slowly changing dimensions is common in ROLAP systems but seldom and poorly supported in cubes.

## SUMMARY OF THE INVENTION

In accordance with an embodiment of the present invention, there is provided a system for storing data. The system comprises one or more member cubes for storing time-partitioned data, and a control cube for accessing the member cubes.

In accordance with another embodiment of the present invention, there is provided a method of transforming a body of data into a time-based partitioned cube. The method comprises the steps of partitioning the data into one or more time-based dimension partitions, creating member cubes corresponding to the one or more time-based dimension partitions, and creating a control cube for representing the data distributed over the member cubes.

In accordance with another embodiment of the present invention, there is provided a method of querying a time-based partitioned cube. The method comprises the steps of analyzing a query received for a body of data organized into a time-based partition cube, redirecting the query to one or more member cubes, and aggregating results received from the one or more member cubes.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows an example of a time-based partitioned cube, in accordance with an embodiment of the present invention.

Figure 2 shows an example of a method for transforming a body of data into a time-based partitioned cube, in accordance with an embodiment of the present invention.

Figure 3 shows an example of a schematic representation of partitioning data in time, in accordance with an embodiment of the time-based partitioned cube.

Figure 4 shows another example of a time-based partitioned cube.

Figure 5 shows another example of a partition of data in time, in accordance with an embodiment of the time-based partitioned cube.

Figure 6 shows another example of a partition of data in time, in accordance with the time-based partitioned cube.

Figure 7 shows a flowchart for a method of using a TB cube for querying and reporting.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Time-based partitioned cubes (TB cubes) are a multidimensional data source used for querying and reporting in the field of online analytical processing. TB cubes involve a process whereby a set of cubes are treated as one large cube. The member cubes are created separately and then pulled together as a larger cube by a control cube. The control cube contains information about the overall structure of the cube (dimensions and measures) and certain category related information that is valid for the entire larger cube (the structure of the time dimension, the expressions for Calculated categories, etc.). The member cubes are distinct from each other in a single dimension. That is, there is a dimension in which the categories in each member cube are distinct from those in all the other member cubes.

A time-based partitioned cube (TB cube) is a potentially large collection of cubes that is seen as a single cube by the user. A member cube is one of constituents of a TB cube. Each member cube is a regular cube – it can be accessed either directly as an independent cube or transparently as a component of a TB cube. Member cubes normally are selected from (a subset of) a cube group partitioned along the time dimension. A cube group is a set of cubes that are generated from a common model by partitioning of the model horizontally (alongside the time dimension) and, optionally, also vertically (by aggregation to some level in a dimension). The amalgamating agent of a TB cube is its control cube -- a cube that contains only metadata about members of a given TB cube – what they are, how they are related to each other on time scale and how they are deployed. Both membership and deployment can be dynamically changed. There is one control cube per TB cube. One purpose of a control cube is to provide the entry point for end users to access the TB cube. Another purpose of a control cube is to acquire dynamically the control data about the current set of member cubes (their mapping onto amalgamated time dimension and their deployment). The MOLAP query engine uses this

information at run time to resolve queries (to decompose and route them to member cubes). Any number of TB cubes can be formed from a given cube group, each with its own control cube. Their memberships can overlap.

Figure 1 shows an example of a TB cube 20, in accordance with an embodiment of the present invention. The TB cube 20 comprises one or more member cubes 22 for storing data partitioned using the time dimension, and a control cube 21 for accessing the member cubes 22. The control cube 21 has an entire time dimension 23 relative to the member cubes 22, a listing of the union of other dimensions 24 that member cubes 22 may have, and a listing of the union of measures 25 that member cubes 22 may have. The control cube 21 may also have a listing of the union of categories that member cubes 22 may have.

The members (member cubes) 22 of the TB cube 20 each cover a distinct time period (such as year, quarter, month, day, etc.). The length of the time period and the granularity of the time dimension does not need to be necessarily the same in all cubes, i.e., one member can cover a year, structured into quarters and months, while some other member can cover a month with details kept at day levels.

For example, a TB cube 20 may span the time interval January 2000 to May 2002, with member cubes 22 for the following five time intervals:

2000

2001

Q1 2002 (first quarter of 2002)

Apr 2002 (April 2002)

May 2002.

Each time interval comprises a member cube 22. The entire time dimension 23 (i.e., the time dimension that spans all five members) may be ragged: all time levels do not need to be instantiated in all members. One possible use of a ragged time dimension 23 is that historic data can be kept aggregated over time while more recent and current data may be kept in detail. Thus, in the above example, the time granularity in cube 2000 may be "month", while in the remaining four cubes (cubes 2001 through May 2002) the time granularity may be "day".

Figure 2 shows an example of a method for transforming a body of data into a TB cube 20 (30), in accordance with an embodiment of the present invention. The method (30) comprises steps of partitioning the data in time (31), creating member cubes 22 (32),

and creating the control cube 21 (33). Once these three steps are performed, the method (30) is done (34).

The first step in the method (30) is to partition the data in time (31). The data is split along its time dimension into a number of smaller partitions. Each partition pertains to a well-defined time interval, such as Week, or Quarter. Figure 3 shows an example of a schematic representation of partitioning data in time (31), in accordance with an embodiment of the TB cube 20. Splits at several levels with different partition sizes of Year 41, Quarter 42, and Month 43, are shown in Figure 3.

Figure 4 shows an example of a TB cube 20 having a control cube 21 and member cubes 22 comprising quarterly time partitions 42. Equidistant partitioning with partition width the same as the period of regular cube update greatly simplifies the update run: The data collected during the update cycle is used to create another member cube, without any affect on older member cubes. With above example, a new member cube would be created at the end of September of current year.

Individual member cubes 22 store both data and metadata that reflect only their respective time period. Therefore a query that is evaluated in any member cube 22 is evaluated in its correct time context and is not affected by data attributes that may have been present in the past (previous cubes) or future (the cubes that follow). Querying member cubes 22 in the chronological (or other) sequence will reflect changes that have occurred in each business dimension 24.

It is not necessary that the time periods be all of the same lengths for all member cubes 22. Figure 5 shows another example of a partition of data in time 60, in accordance with an embodiment of the TB cube 20. Assuming that the current month is August, the time partitions are:

Year 1;
Year 2;
...;
Year X (where X is two years before the current year);
Quarter 1 of Previous Year;
Quarter 2 of Previous Year;
July of Previous Year;
August of Previous Year;
September of Previous Year;

October of Previous Year;
November of Previous Year;
December of Previous Year;
January of Current Year;
February of Current Year;
March of Current Year;
April of Current Year;
May of Current Year;
June of Current Year;
July of Current Year; and
August of Current Year.

The above scheme could be used when the emphasis on recent data is much higher than that on historical data.

If only recent data is to be analyzed in full detail, then it may be advantageous to store historical data in more compact, pre-aggregated form, with coarser time granularity. Widening the time interval for historical data combined with change in time granularity (as in the example above) then serves to balance the individual member cube 22 sizes and, therefore, their performance. Aggregation of older data also helps to maintain the overall on-disk size of a whole TB cube within reasonable limits.

This consolidation of historical data can be part of periodical update: At specific points of time (such as end of each quarter or year end) certain older member cubes 22 can be aggregated into a single cube, with coarser time granularity, if desired.

Figure 6 shows another example of a partition of data in time 65, in accordance with the TB cube 20. The partition is based on a sliding time window. In this example, only new data is of interest. The TB cube 20 consists of, say, 12 monthly cubes 22 for the most recent months. Every time when a new month (say August) is added, the oldest one (here, August Previous Year) is dropped.

Once the data is partitioned in time (31), member cubes 22 are created (32). A set of partitions is selected that covers the time span of data contiguously and then a cube is created from each selected partition. This process yields what will become the set of member cubes 22 of a TB cube 20. For instance, the member cubes 22 could arise from partitioning by Quarters 42, as schematically represented in Figure 4.

Once the member cubes 22 are created (32), a control cube 21 is created (33). In the example of a TB cube 20 shows in Figure 4, the TB cube has a control cube 21 and member cubes 22 comprising quarterly time partitions 42. The control cube 21 serves as the physical entry point to a TB cube: it is the agent that represents, to the consumer, the data that is distributed over member cubes 22 as one single whole. More specifically, the control cube 21 is used:

- to represent to the outside word the data stored in member cubes 22 of the entire cube 20;
- to handle queries against the entire cube 20 – analyze, decompose, route them to appropriate member cube(s) 22 and aggregate the partial results to be returned to its consumer; and
- To keep track of member cubes 22 and their deployment when the composition of a TB cube 20 changes (i.e., when member cubes 22 are added to, or consolidated within, or dropped from, the set).

By designing specific control cubes 21, more that one TB cube can be defined using the same pool of member cubes 22.

A TB cube 20 may have the entire time dimension 23 comprised of adjacent, non-overlapping and, possibly, non-equidistant time dimensions of its member cubes 22. The entire time dimension 23 may be ragged and partial non-conformity of time dimension between member cubes 22 is acceptable. For example, a member cube 1998 can have time rollup levels "year", "quarter", "week", and "day", while a member cube 1999 can have levels "year", "quarter", "month", and "day". However, non-conformity of time dimension between member cubes 22 may have some consequences. In the above example, a query "set of descendants of All Years at level = Month" will yield a result set with no descendants from cube 1998. Also, a pair of relative time categories, such as "current month" and "last year current month", cannot be defined between two such years.

The corresponding objects (such as categories, dimensions 24, and measures 25) of member cubes 22 have the same identification (ID) number. Preferably, the attributes (texts, codes, etc) of corresponding objects are identical (so that the metadata query can be routed to any cube that contains the object). However, not all the objects need to be present in all member cubes 22 and the category rollup hierarchies need not be the same.

In other words, the dimension 24 of member cubes 22 need not to be (exactly) conforming. For instance, if an employee A moves from Boston to New York, it is acceptable to move A's category in the hierarchy: it could be a child of Boston in January cube but child of New York in February cube. A ***getChildren*** function executed against the TB cube 20 would yield different results depending on the current time context: Boston children would include A in January or Q1, but not in February or March. A query

Sales (Q1, North America)

would include A's contribution made in both Boston and New York, while

Sales (Q1, Boston) or Sales (Jan, North America)

would include only the employee's Boston contribution.

The TB cube 20 support of this feature (known as slowly changing dimensions) can be applied not only to moving categories about hierarchy but also to things like discontinued products (or introducing new products), Company reorganization, etc. The fact that metadata queries will show only those categories that are applicable in given time context may significantly reduce the frequency of N/A cells and rows in reports.

The member cubes 22 that comprise a TB cube 20 are normally selected from a cube group – one that is generated by a database modeling tool from a common model. In such case, the dimensions of member cubes 22 will be conforming; with only exception that not all children of a category may be present in each cube 22. In each cube 22, only those categories are instantiated for which some transactions (or, datapoints) have occurred in given time period.

For example, if a category "Screw" has children "Nuts" and "Bolts" and the TB cube 20 consists of members

Jan, Feb, Mar and Apr

and

there were no sales of Bolts in January

there were no sales of Nuts in February

there were no sales of either Nuts or Bolt in April

then Screw will have both children – Nuts and Bolts – only in Mar cube. In Jan cube, the child Nuts will be missing. In Apr cube, the whole subtree {Screw, Nuts, Bolts} will be missing.

It is not required that all dimensions 24 exist in all member cubes 22. For example, if the Company introduced tracking of sales channels in 2001, the member cube 22 for year 2000 will have Sales Channels dimension missing. Similarly, not all measures 25 need to exist in all member cubes 22. Furthermore, measures do not need to be in the same order (and organized into the same folders) in all member cubes 22.

The control cube 21 does not store data. As is described above, the control cube 21 serves as entry point (proxy) for the whole TB cube 20. To access the time based partitioned cube 20, a user opens (connects to) the control cube 22 (e.g., using a .mdc file). The control cube 21 also serves as a container for the scaffolding metadata for whole TB cube 20, namely:

- how the time dimension 23 is partitioned (i.e., the hierarchy of time dimension 23 above the roots of member cubes 22);
- the complete list of all dimensions 24 that exist in member cubes 22; and
- the complete list of all measures 25 (and the structure of measure folders, if any) that exist in member cubes 22.

The control cube 21 builds dynamically the control information used by a query engine:

- to route queries (both metadata and data queries) to individual member cubes 22; and/or
- to decompose queries whose time interval spans several member cubes 22, dispatch them and aggregate the partial results before passing the query result back to the user.

Metadata stored in control cubes 22 are described below.

There is a dimension record for every dimension 24 that exists in a member cube 22. As mentioned above, except for the Time 23 and Measure dimensions, not all dimensions 24 need to be present in all member cubes 22. All dimensions 24 except the partitioning (Time) dimension 21 and Measure dimension contain only a single category – the dimension root. The order of the dimension 24 that is specified in a control cube 20 is the order in which the dimensions are presented by a ***getDimensionList*** call against connection to (the control cube 20 of) the TB cube 20. The order may differ from that returned by a ***getDimensionList*** call for a connection to a specific member cube 22.

There is a Measure record for every measure 25 that exists in a member cube 22. As mentioned above, some measures 25 and / or measure folders can be missing in some

member cubes 22. A Measure dimension structure as defined in the control cube 21 is the one that is presented to the user in a ***getMeasureList*** call and a ***getChildren*** call against measure dimension.

The time dimension 23 is the partitioning dimension in a TB cube 20. The time dimension 23 in a control cube 21 describes the time interval that spans at least the time intervals of all member cubes 22 and, potentially, it encompasses also other (past or future) time intervals. The granularity of time in control cube 21 may, but need not, correspond to that in member cubes 22 – it may be either coarser or finer. Also, the hierarchy may be ragged. The only condition is that the root categories of time dimensions 23 of member cubes 22 must be present in the time hierarchy of control cube 21. Any number of alternate time hierarchies can also be predefined in control cube 21, in terms of time categories of its primary hierarchy. Both the mapping of time hierarchies of member cubes 22 onto the hierarchy (or hierarchies) predefined in control cube 21 as well as the resolution of non-conformity in time span and granularity are dynamic; based on control data (TB cube's 20 current membership and its deployment) acquired by MOLAP query engine dynamically, at run time.

Each member cube has internally assigned a unique ID: member_no. This is a number between 1 and N (N = number of member cubes 22) assigned in reversed chronological order. In the January 200 to May 2002 example described above, the assignment would be

| *Member_no* | *member cube* |
|---|---|
| 5 | *{2000 cube file address}* |
| 4 | *{2001 cube file address}* |
| 3 | *{Q1 2002 cube file address}* |
| 2 | *{Apr 2002 cube file address}* |
| 1 | *{May 2002 cube file address}* |

Queries that are issued against a control cube 21 are dispatched to member cubes 22, depending on the time dimension 23 category specified for the query, to one or more member cubes 22. The facility that handles decomposition, dispatch and reassembling the query results is referred to as the MOLAP query engine. There are two controls maintained by the query engine and utilized by it to route queries: current time context and time category membership table.

Time category membership table is a dynamic table. It is maintained by the query engine in a database application internal cache of the control cube 21. The time category membership table is created at the time of a control cube 21 Open and it is discarded by its Close.

The table contains one row for each partitioning (time category) that has been encountered (referenced) during the Open / Close session. There are three columns (8 byte per record):

- Category ID (the unique key)
- Member_no of the cube where this category exists (for example, the category Apr 2001 will have member_no = 4, Apr 2002 will have member_no = 2
- Properties: list of flags utilized by the query engine when dispatching (such as "Is the category the root of a member?", "Does the category exist on alternate path only?", "Are the children of this category already loaded into the table?", etc.)

The current time context is a control derived from the time category (or the set of time categories) that were included in the context list of categories specified in the encompassing high-level query. To route a query,

- the current time context prevailing at the time of query is augmented by the time context from the query call (such as domain list specified in data query, or parent category for GetChildren, etc).
- For each category from the resulting time context the query engine determines the member_no (or list of member_nos) of cubes that the category belongs to. The source of this information is time category membership table and/or the hierarchy of control file time dimension. For example, if the resulting query context contain categories Q2 2002 and Apr 2002, the member_no lists will be

| Category | Member_nos |
|---|---|
| Q2 2002 | 1, 2 |
| Apr 2002 | 2 |

The query is routed to the intersection of all member lists for the resulting time context (cube member_no = 2 in above case).

Advantageously, the TB cube 20 allows for better performance when updating a cube on a regular interval (say weekly or monthly). The TB cube 20 also allows for improved performance when accessing such a cube. Better scalability is also provided with the TB cube 20. Larger data sets can be handled and the performance is better. Furthermore, metadata time dependency is intrinsic to a TB cube 20.

The concept of time partitioning of raw data (and identification of partitions that cover the time span of interest) is inherent in the TB cube 20. The ability to convert data from selected partitions into independent member cubes 22 (with aggregation and time granularity set as appropriate) and the ability to add and drop member cubes 22 dynamically (with no effect on the rest of the set) is also provided by the TB cube 20.

The TB cube 20 also comprises a routing algorithm that is able to present and handle the set of independent cubes as if it were a single big cube. The routing algorithm analyzes queries against the data (or metadata) stored in member cubes 22, dispatches sub-queries to appropriate member cubes 22 based on the time context and consolidates the partial results. As indicated, the routing scheme adapts itself dynamically to changes in the composition of the member cube 22 set as well as variances in time granularity of individual members 22.

The TB cube 20 provides the ability to handle dynamically changes in data aggregation and/or its time granularity in any portion of time scale. The TB cube 20 also provides the ability to implement simple security schema: different control cubes 21 over the same pool of member cubes 22 can restrict the data access to different portion of data for different users.

Figure 7 shows a flowchart for a method of using a TB cube 20 for querying and reporting (70). The method (70) begins with the TB cube 20 receiving a query from a user (71). The user query is intercepted by the control cube 21. The control cube 21 analyzes the user query (72) and determines which member cube 22 to redirect the query (73). The member cubes 22 return the results (i.e., partial results) from the redirected queries to the control cube 21 (74). The control cube 21 aggregates the partial results into a final result for the user (75). The method (70) is done (76). Other steps may be added to the method (70).

The logic described in the flowchart shown in Figure 7 may be added to a query engine, such as a MOLAP query engine, to handle queries made to TB cubes 20. The user does not know (and does not need to know) whether the query has been evaluated in

a TB cube 20 or a regular cube. The user can the put the result into an appropriate cell of a report without knowing (and caring) where the value came from. Thus, the use of a TB cube 20 is transparent to the user.

Specific advantages to the TB cube 20 include scalability, updatability, expanded functionality and run time performance.

Scalability. The size of a TB cube 20 is theoretically unlimited. A TB cube 20 can comprise any number of member cubes 22 (practically perhaps up to several thousands); each of them 22 can be a huge regular cubes. A TB cube 20 implementation may thus solve one hurdle that is experienced with cubes: the inability to accommodate unlimited data.

Updatability. The incremental update of a TB cube 20 will amount to simply adding one more member cube 22 to the TB cube 20, without any need to re-build the remaining member cubes 22. This sharply reduces both the update time requirements of current practice (where the whole cube must be rebuilt) and the dangers of instability introduced by such update. Also, the diminishing importance of historic data is easier to handle: It would be much easier (and faster) to drop from a TB cube 20, for instance, three (daily details) member cubes 22 for Jan, Feb and Mar of last year and replace them by one (consolidated) member cube 22 for Q1 than to perform similar update of an equivalent regular cube. Removing / archiving irrelevant old data from a TB cube 20 amounts again just to dropping of a member 22 from the cube 20 (that is accomplished by simple editing of a definition file).

Expanded functionality. External partitioning in time dimension 23 together with allowing non-conformity of other dimensions 24 introduces support of slowly changing dimensions into cubes.

Run time performance. The deterioration of a TB cube 20 performance as compared with performance of an equivalent regular cube is roughly proportional to the number of member cubes 22 that the query (either metadata or data) has to be decomposed into. A query routed to a single smaller member cube 22 should execute much faster than in an equivalent regular cube (shorter bitmaps to handle). This may be beneficial also for decomposed queries: the overhead of routing and aggregation will at least partially be counterbalanced by the gains from queries to smaller cubes.

Improvement in zero-suppress performance can be expected in reports generated for specific (narrow) time (interval). The categories that normally yield empty rows (or

columns) in some time context in regular cubes will not be even available for reporting in TB cube 20 in that time context. In other words: A report will be generated without the categories that do not exist in the time interval, so there will be no need to suppress them.

The TB cube 20 according to the present invention may be implemented by any hardware, software or a combination of hardware and software having the above described functions. The software code, either in its entirety or a part thereof, may be stored in a computer readable memory. Further, a computer data signal representing the software code which may be embedded in a carrier wave may be transmitted via a communication network. Such a computer readable memory and a computer data signal are also within the scope of the present invention, as well as the hardware, software and the combination thereof.

While particular embodiments of the present invention have been shown and described, changes and modifications may be made to such embodiments without departing from the true scope of the invention.

WHAT IS CLAIMED IS:

1. A system for storing data, the system comprising:
one or more member cubes for storing time-partitioned data;
a control cube for accessing the member cubes.

2. The system as claimed in claim 1, wherein the control cube has:
an entire time dimension relative to the member cubes;
a listing of other dimensions of the member cubes; and
a listing of measures of the member cubes.

3. The system as claimed in claim 2, wherein member cube are removed from the system.

4. The system as claimed in claim 2, wherein member cubes are added to the system.

5. The system as claimed in claim 2, wherein the control cube restricts access to member cubes.

6. The system as claimed in claim 2, further comprising a plurality of control cubes, each control cube coupled with a group of member cubes from a pool of member cubes to form a separate time-based partitioned cube.

7. The system as claimed in claim 6, wherein different control cubes over the same pool of member cubes restricts data access to different portions of data for different users.

8. A method of transforming a body of data into a time-based partitioned cube, the method comprising the steps of:
partitioning the data into one or more time-based dimension partitions;
creating member cubes corresponding to the one or more time-based dimension partitions; and
creating a control cube for representing the data distributed over the member cubes.

9. The method as claimed in claim 8, wherein the data is partitioned into equidistant time intervals.

10. The method as claimed in claim 8, wherein the data is partitioned into non-equidistant time intervals.

11. The method as claimed in claim 8, wherein the data is partitioned into a sliding window of time intervals.

12. A method of querying a time-based partitioned cube, the method comprising the steps of:

analyzing a query received for a body of data organized into a time-based partition cube;

redirecting the query to one or more member cubes; and

aggregating results received from the one or more member cubes.

13. An online analytical processing query engine comprising a logic module for implementing the method of claim 12.

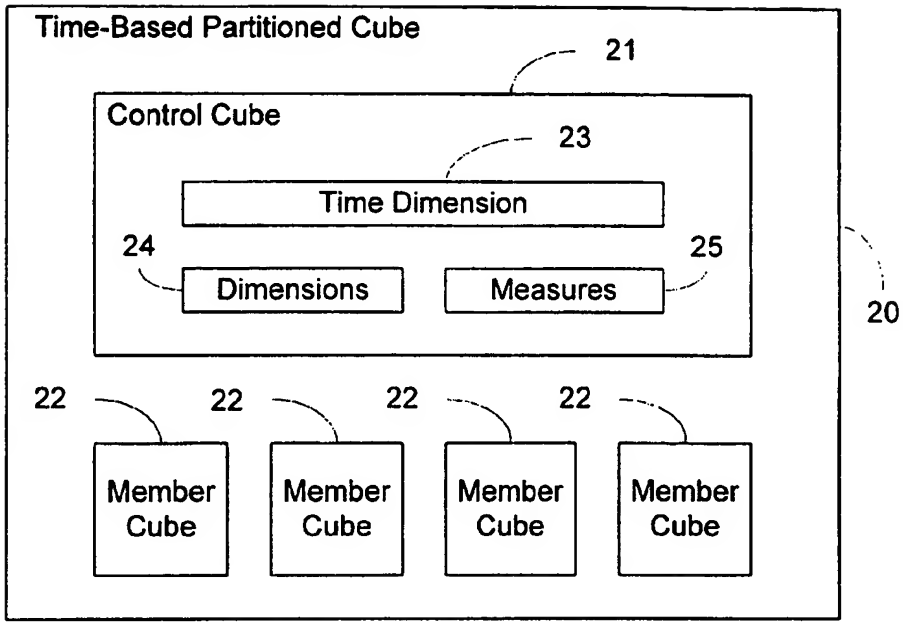


Figure 1

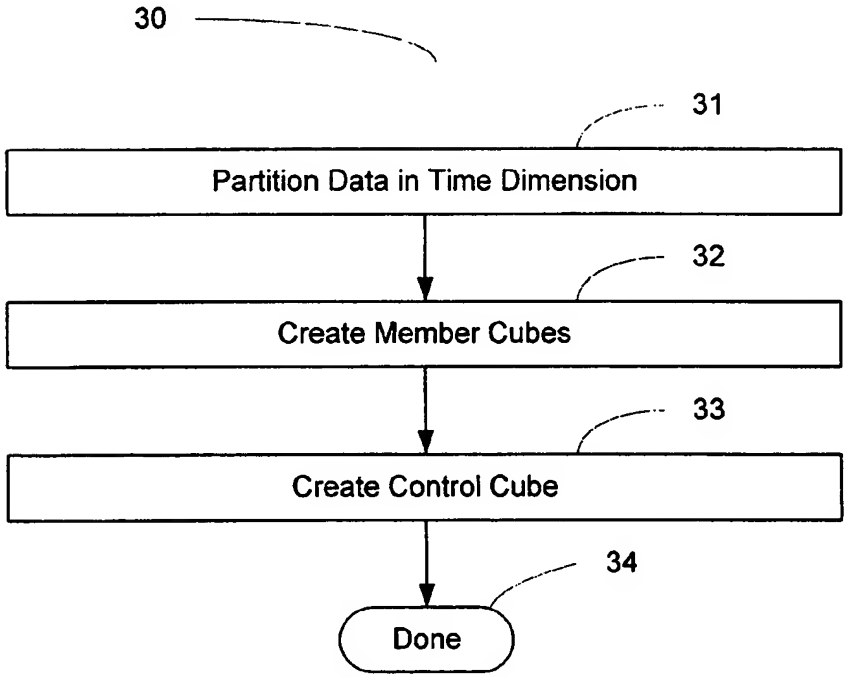


Figure 2

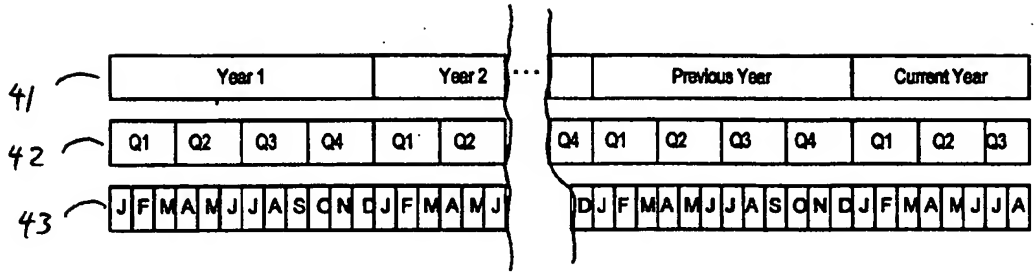


Figure 3

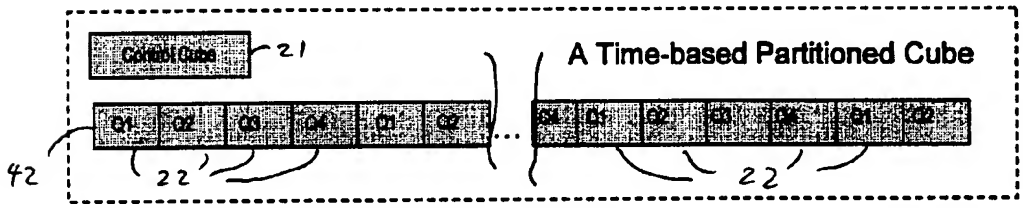


Figure 4

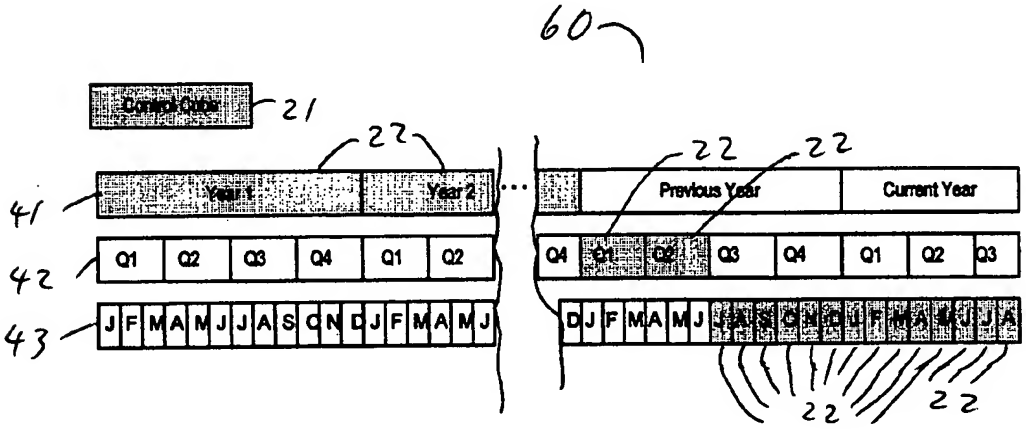


Figure 5

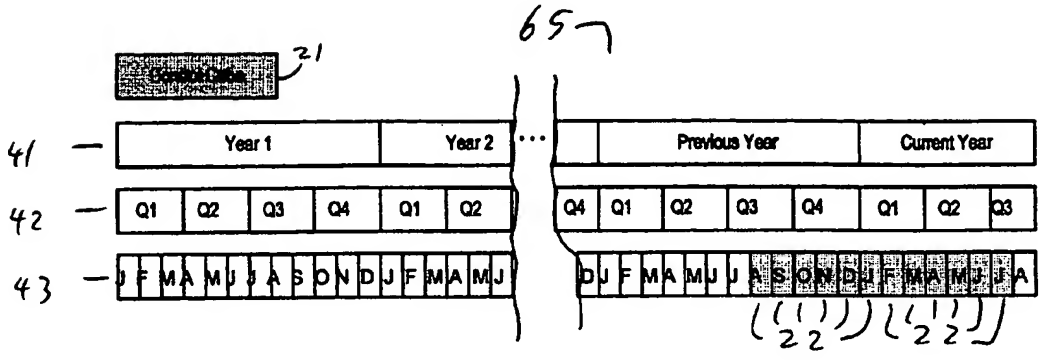


Figure 6

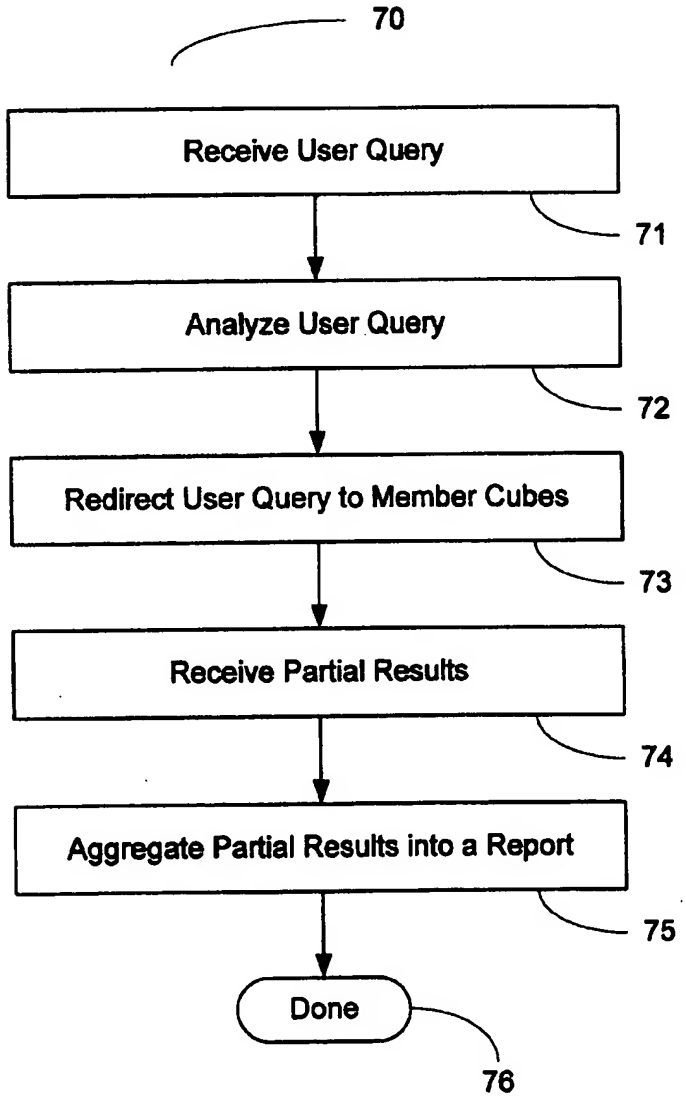


Figure 7